

Music Runs With Me

Harry Chung

MUMT 306
Music & Audio Computing I

Class Project

1 Motivation and Objective

Rhythmicity exists in the physical and biological worlds, from cellular pacemakers to human locomotion. At the large biological level, rhythmic movements are controlled by primitive neural environment called Central Pattern Generator, or CPG. This is a collection of biological neural circuits that produces rhythmic outputs in the absence of rhythmic input. [3] Such mechanism allows a runner, for example, to operate muscular system to operate at a consistent mechanical output without using much brain power. In order to maintain this state, there must be minimal rhythmic input, especially the one that interferes with the natural physical rhythmic output. Dissonance between the frequency of the physical rhythmic movement and modulation frequency (eg BPM of music) detected from auditory sensory could influence CPG, and thus disrupts one's efficiency of carrying out the cyclic task.

The goal of this project is to create a working MAX/MSP patch and Arduino code that detect the cadence of one's rhythmic and cyclic movement, and rearranges and modifies a bank of randomly selected music to match the cadence. This process is done in three steps:

1. Cadence detection,
2. BPM extraction,
3. Playlist rearrangement.

This project is meant to explore the design idea, and provide design aspects that are worth considering if it becomes a viable commercial product. Hence, the focus of this study will be practicality and viability, rather than optimization. At a personal level, this study is used to practice MAX/MSP.

2 Methodology

2.1 Cadence

In running, cadence is defined as number of steps taken per minute. This is generally considered the only practical control variable for runners to increase/decrease speed. While there's no magic number, it's highly personal and an important training consideration. Most recreational runners maintain cadence of somewhere between 150spm and 170spm.[1]

2.2 Hardware

There are multiple ways to detect the runner's cadence. In this study. I used a high precision 6 axis gyroscope, MPU6050, for prototype purpose combined with a simple arduino board as shown in Figure 1:

The electronics were mounted on a breadboard to mimic a shoe. However, it is worth noting that this device can be installed anywhere on the body that also exhibits periodic motion, such as arms.

This set up produces the sensor's angle in degrees relative to the flat, horizontal surface. For this application, I was interested in only the angle of the shoe from the ground. The extraction code was based on Jeff Rowberg from Github [2].

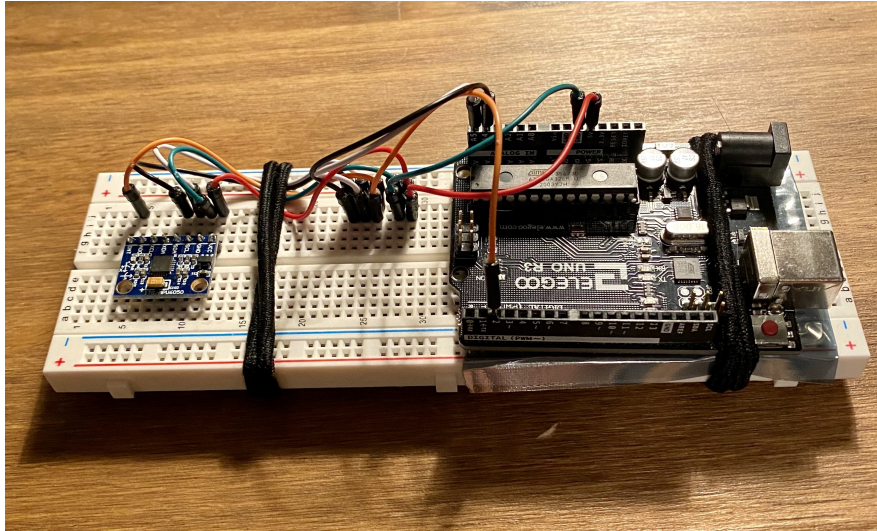


Figure 1: MPU6050 with Arduino

2.3 Calculation

Rather than measuring the elapsed time between two predetermined position, the MAX/MSP calculates the cadence based on the change in foot angle for each predetermined time duration, or $\Delta\theta/\Delta t$. As mentioned earlier, cadence is expressed in steps per minute. This can be reworded as rotation per minute, where one rotation is one stride of the periodic feet movement. Hence,

$$\text{Cadence} = \frac{\Delta\theta}{\Delta t} = \frac{\Delta\theta}{10 \text{ ms}} \cdot \frac{60000 \text{ ms}}{1 \text{ min}} \cdot \frac{1 \text{ rotation}}{360^\circ} = \# \text{ of rotations per minute} \quad (1)$$

To minimize the micro variation, the patch saves the “instantaneous” cadence measurements over one second, and calculates the average. Then, it sends out the average data every two seconds. This data output frequency can be adjusted later on. If the goal is to change the music playback speed as frequently as possible, then such high frequency data feedback is necessary. However, considering the stable periodicity of experienced runners, frequent change in music playback speed may disturb the runner. Hence, it may not even necessary until the end of the track. For the sake of quick in-class demonstration, I had chosen two seconds.

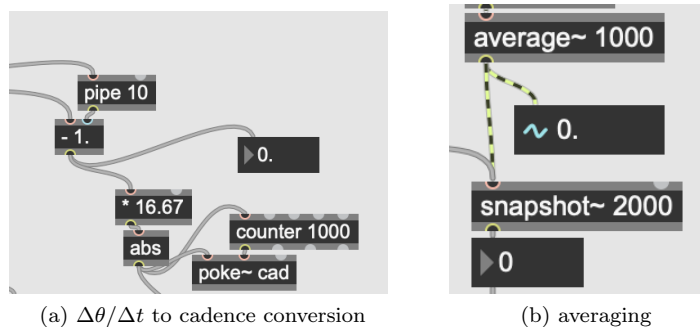


Figure 2: Cadence patch

2.4 BPM Extraction

For this project, I decided to use Midi files. The core idea is the same for any music files, although the process of extracting the BPM data will differ. In the case of Midi file, the BPM data is indirectly contained in the file's meta message, also known as tempo message. This tempo message is in the unit of microseconds per quarter note.

By design, the file sends out 255 (or 0xFF) if it's sending out a meta message. Then, if it's a tempo message, it sends out 81 (or 0x51) as its second byte. The next byte is indicates the number of bytes to represent the actual tempo. After concatenate the tempo message bytes, it will need to get converted to beat per minute (BPM). The patch is essentially a message filtering scheme to efficiently extract the tempo message.

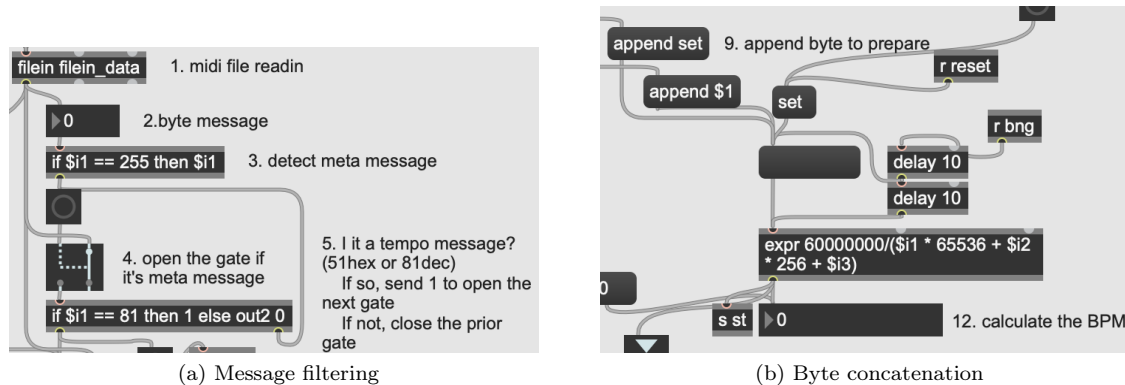


Figure 3: BPM extraction

It is worth noting that the BPM extraction stops after it detects and sends out the first tempo message. Music files typically contain multiple tempo messages, as the music goes through different tempo changes. It is rare for typical “workout” to go through dramatic changes in tempo. Furthermore, constant change in playback speed could disturb the listener. However, It is worth looking into the possibility of picking the right tempo, as the first tempo may not be the best tempo or average tempo through out the music.

2.5 Playlist Rearrangement

The initial Midi bank was created by randomly placing files into a patch as shown below:

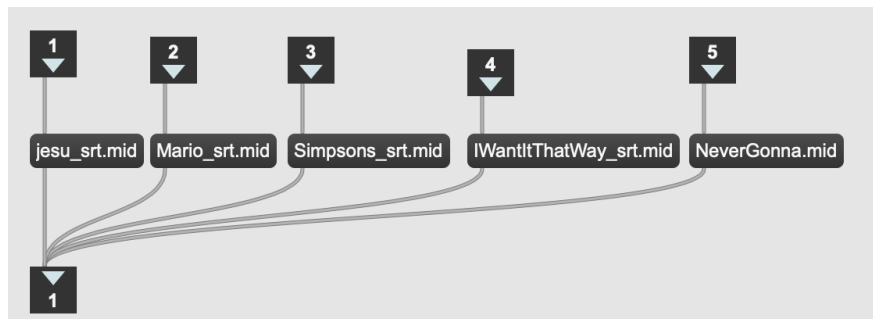
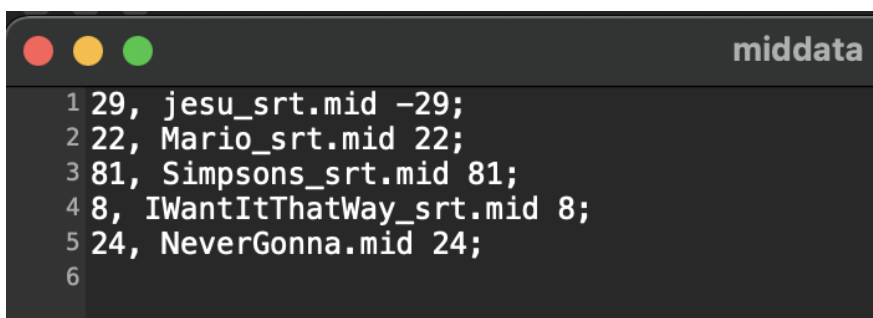


Figure 4: Bank of music

The main patch goes through each file in the bank and then extracts the first BPM via aforementioned

BPM extraction patch. The BPM is then subtracted from the cadence. This calculation gives each track's temporal relevance to the running cadence. Collected data is saved by *coll* object as shown below:



```

1 29, jesu_srt.mid -29;
2 22, Mario_srt.mid 22;
3 81, Simpsons_srt.mid 81;
4 8, IWanItThatWay_srt.mid 8;
5 24, NeverGonna.mid 24;
6

```

Figure 5: Collected list

The first number in the list is $|BPM - Cadence|$, which represents scalar offset between the cadence and the BPM. This will be the reference number when the list gets rearranged based on the tempo relevancy. The second number shows the direction, which indicates whether the BPM is too slow or too fast compared to the running cadence. By finding the percentage difference between BPM and cadence, the patch adjusts the playback speed as shown below. “1024” message instructs the *seq* object to play at normal speed, so adjusting “1024” by the percentage gives the appropriate playback speed message.

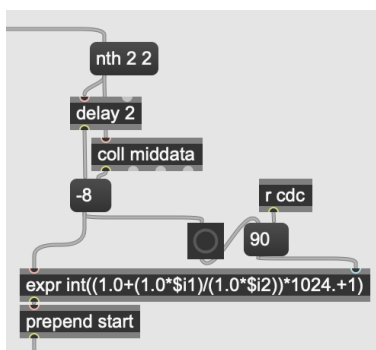


Figure 6: Playback speed modification

Once the playback speed and the order is determined, then the patch extract the second data from the list, and feeds into the *seq* object to play the music. Once the track is done, it removes the first track in the list, and repeats process to the next closest track with appropriate playback speed modification.

3 Challenges

The most frequent challenge I have faced during the project was figuring out ins and outs of different MAXMSP objects. There are countless objects that weren't even mentioned in class, and not knowing what exists limited my creativity in creating the main patch. I am certain that there are many ways to make the patch more efficient, and perhaps enough to be used in small, wireless devices. At the same time, the purpose is to initialize the idea, so I will consider this as a rough R&D phase. Another challenge was making manual decisions. For instance, I had to manually decide that 2-steps-to-1-beat ratio is ideal for this application. It will be worth investigating what the actual ideal ratio is and if there is a way to automatically adjust depending on the music.

4 Potential Usage

This study is focused on practicality of commercial application. Most modern electronics already contain gyroscopes that are plenty accurate enough for measuring and calculating cadence. Along with Spotify's developer platform, which provides audio analysis, including the tempo, and portable devices, such as Garmin or AppleWatch, it should not be a complicated product that many find to be useful. Similarly, any periodic biological signal can be used. For instance, heart beat is a common periodic biosignal. One could develop use this indicator to rearrange or modify music for meditation.

References

- [1] A. Burdick. Finding your perfect run cadence, Oct 2016. <https://www.trainingpeaks.com/blog/finding-your-perfect-run-cadence/>.
- [2] J. Rowberg. jrowberg/i2cdevlib. <https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050>.
- [3] I. Steuer and P. Guertin. Central pattern generators in the brainstem and spinal cord: An overview of basic principles, similarities and differences. *Reviews in the Neurosciences*, 30, 12 2018.